Decal Actors

A guide to using the Deferred Decal actor.

Deferred decals offer better performance and easier maintenance. Writing to the GBuffer instead of recalculating lighting has several benefits:

- The performance with many lights gets much more predictable because there is no limit on the light count or type as the same code path is used for all of them.
- Manipulating a screen space mask also allows effects that otherwise would be considered difficult (e.g. wet layer).

The decal is rendered by rendering a box around the decal affecting area.

Adding decals to the level

The easiest way to add decals to a scene is to select an appropriate decal material in the **Content Browser**, then **right-click** inside the Viewport and choose **Place Actor** from the context menu. The decal may then be resized and oriented using the transformation tools.

Sizing, tiling and offsetting

Once the decal is created, it can be positioned and oriented using the translation and rotation widgets.

The non-uniform scaling widget controls the width, height, and far-plane distance of the decal volume.

Deferred Decal Properties

A deferred decal has only a couple of properties:

Item	Description
Material	This holds the material that will be used as a decal.
Sort Order	This allows the user to set a value to control how multiple decals will sort when stacked. Higher values render on top.

Be careful when setting sort values. Setting too many sort values on too many different decals prevents those decals from being sorted via state, which can harm performance.

Material Settings

The **DecalBlendMode** setting defines how the material properties (diffuse, specular, normal, opacity, ...) are applied to the GBuffer.

The opacity is used to blend the decal contribution. An efficient decal is manipulating only few GBuffer properties. The cases we currently optimize for are represented by the other modes: ***DBM_Diffuse***, ***DBM_Specular***, ***DBM_Emissive***, ***DBM_Normal***.

Item	Description
Translucent	This will blend the full material, updating the GBuffer, and does not work for baked lighting.
Stain	This is Modulate BaseColor, blend rest, updating the GBuffer, and does not work for baked lighting.
Normal	This will only blend normal, updating the GBuffer, and does not work for baked lighting.
Emissive	This is for Additive emissive only.
DBuffer_Translucent Color, Normal, Roughness	This is for non-metal, put into DBuffer to work for baked lighting as well. This will become DBM_TranslucentNormal if normal is not hooked up.
DBuffer_Translucent Color	This is for non-metal, put into DBuffer to work for baked lighting as well.
DBuffer_Translucent Color, Normal	This is for non-metal, put into the DBuffer to work for baked lighting as well. This will become DBM_DBuffer_Color if normal is not hooked up.

Item	Description
DBuffer_Translucent Color, Roughness	This is for non-metal, put into DBuffer to work for baked lighting as well.
DBuffer_Translucent Normal	This is for non-metal, put into DBuffer to work for baked lighting as well.
DBuffer_Translucent Normal, Roughness	This is for non-metal, put into DBuffer to work for baked lighting as well. This will become DBM_DBuffer_Roughness if normal is not hooked up.
DBuffer_Translucent Roughness	This is for non-metal, put into DBuffer to work for baked lighting as well.
Volumetric Distance Function (experimental)	This will output a signed distance in Opacity depending on LightVector.

DBM_ScreenSpaceMask is special as it affects a special masking channel which is currently used by SSAO (Ambient Occlusion). This allows the decal to override or fade the contribution in some areas.

DBM_DiffuseSpecularEmissive is the mode that affects multiple GBuffer channels.

Note that the material blend mode also affects how the GBuffer values are blended together. So it is possible to multiply the diffuse color.

You can use the GBuffer view mode to inspect the GBuffer values that are stored per pixel.

The decal local position is a 3d position in the range 0 to 1. The texture UV is giving you the x and y component. In case you need the z component, you can hook up a CameraVector node to get all 3 vector components.

Performance

The mesh complexity of the objects affected by the decal is not affecting the performance. The decal performance depends on the shader complexity and the shader box size on the screen.

We can further improve the per decal performance. Ideally the bounding box of the decal is small to get better per pixel performance. This can be done manually. An automated method is possible but a good designer can also adjust the placement to improve performance further.

The view mode **ShaderComplexity** (editor UI or "viewmode ShaderComplexity") can be used to see the impact on the pixel shading cost, it uses a pixel shader cost estimate and accumulates where multiple decals overlap. At the moment the decal masking feature has no effect there (masked parts should have a small constant cost because of using the stencil hardware feature). The following shows a scene without (left) and with decals (right), in normal rendering (top) and with the ShaderComplexity enabled (bottom):

The darker color indicates a higher performance cost of those pixels. This information can be used to optimize the right shaders, remove barely visible decals or place them more efficiently.

Current limitations

- We currently only support deferred decals and they only work on static objects.
- Normal blending is currently not wrapping around the object.
- Streaming is not yet hooked up so make sure the texture is not streamed.
- Masking decals (not affecting other object) is not fully implemented.

The 2×2 block artifacts fix

Decals may have 2×2 pixels block artifacts on edges as shown in the screenshot below.

This is where the node "Decal Derivative" comes in. This node must be used carefully as it has a very big performance impact. It returns the derivatives on the X and Y axis of the decal's default texture coordinates necessary for anisotropic texture filtering, but computed differently than using the hardware's default and DDX/DDY nodes, to avoid this 2×2 pixel block artifact.

And using it fixes the artifacts on the decal that benefits from it.

Current limitations

- DecalMipmapLevel doesn't support custom UV, but you can patch up its output.