Decal System Reference

Version 1.3

by

1. Introduction

Decals in game engines are images or certain sections of images that are projected onto surfaces. The Decal System implements exactly that for the Unity game engine.

We tried to make the whole solution as intuitive and easy to handle as possible, while still being highly configurable. It was built with Unity's Lightmapping, Level of Detail and Prefabs in mind and contains special solutions to support all of them.

The Decal System creates meshes to visualize the actual decals. If several projectors are used within a decals instance, only one mesh is created to keep the draw calls as low as possible. This is certainly only true as long as the produced mesh contains less vertices than the maximum supported amount which is given by Unity. Otherwise more than one mesh is being generated.

REMARK

The mesh data from projectors of the same decals instance is automatically combined.

1.1. Terminology

(Skinned) Decals

You can understand decals as the central management object. Each decals may have several projectors and thus would consist of more than one decal. That's the reason why we named it in the plural form.

It is in a parent game object relative to the game objects which contain its projectors. Further, it creates and destroys decals mesh renderers and contains the uv rectangles.

Component type: *DS_(Skinned)Decals*

UV Rectangle

The material for the decals may contain a texture atlas that has to be splitted. Each of those parts which may be projected needs to be defined in its own uv rectangle.

(Skinned) Projector

Game objects with projectors can be moved, rotated and scaled in the scene view to achieve the

desired projection. It is further possible to specify which surfaces have to be affected.

Component type: *DS_(Skinned)Projector*

(Skinned) Projector Group

Groups are used to bundle several projectors. This allows you to change the transform of those projectors at once, without loosing the inspector view.

Component type: *DS_(Skinned)ProjectorGroup*

(Skinned) Decals Mesh

The generated meshes which are placed in decals mesh renderers.

Component type: *DS_(Skinned)DecalsMeshRenderer*

2. Getting Started

Creating a new (skinned) decal is straight forward. With the Decal System package imported, you need to select **GameObject → Create Other → (Skinned) Decals** from the menu.

3. Decal Inspector Modes

The configuration of (skinned) decals happens within the inspector view. At the top you see a toolbar with four modes. We are going to explain all of them shortly. A more detailed description of each mode can be found in the following sections.

Decals, projectors and groups share the same inspector view. This behavior was introduced to increase the usability. Only the projectors have a special inspector view for their configuration.

Decals Mode

Lets you specify some high level properties for the mesh that is created, such as the material and projection type.

UV Rectangles Mode

To keep the draw calls low, textures are frequently bundled in texture atlases. The splitting of them happens in this mode and the corresponding window.

Decal Projectors Mode

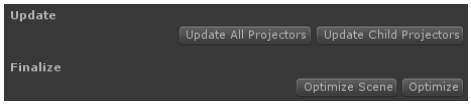
Projectors are configured in this mode.

Settings Mode

Some general settings can be modified in this mode.

3.1. Update and Finalize

In the first three modes you find the update and finalize buttons at the end of the inspector view.



The update buttons either update all projectors or only the currently selected one with all its children.

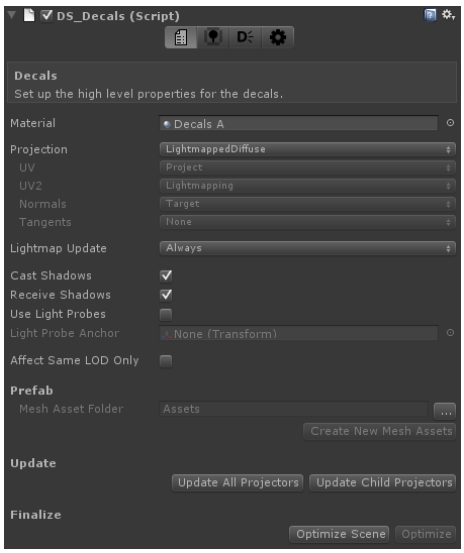
Finalizing the decals is useful to get the best possible performance for the decals. You can optimize the selected decals instance or all decals from the scene at once. This is especially useful if no more changes are being made in the level or just before baking the lightmaps.

REMARK

Optimize the decals in your level to improve the performance.

3.2. Decals Mode

In this mode several high level properties can be specified.



Material

Select the material that has to be used for the generated mesh or meshes. The Decal System is bundled with several Shaders that can be used in the material. In the case that a texture atlas is assigned in the material, the splitting into the individual parts can be handled in the UV Rectangles Mode.

Quality

Only Skinned Decals have this option. You can specify a quality for each skinned mesh with which the skinning computation in Unity takes place. This value is passed to all the skinned mesh renderers that are generated by this skinned decals instance.

Projection

The projection mode defines which mesh data has to be used or needs to be calculated. Several predefined options are available. If you want to specify the projection details on your own, you need

to choose the *Advanced* option.

UV

The primary uv channel of the generated decals mesh or decals meshes can not be empty. Those coordinates are usually *Project*ed, meaning they are computed relative to the projectors. Another option is to use the uv's or uv2's from the target, in which case the coordinates are taken from the surface onto which the projection is being made. If you select *TargetUV* or *TargetUV2*, but the surface does not have the required data, no projection is being made. Terrains are an example in which a projection would fail, if the target's uv's or uv2's are needed, because terrains have neither of them.

UV2

Contrary to the uv's, the uv2 channel in the generated decals mesh or decals meshes is optional. It can also be projected or may reused some existing uv or uv2 coordinates from the target. Additionally there is an option for lightmapping. More information about that can be found in the Lightmapping section.

Normals

You have only two alternatives for the normals. Either they are not used, or the ones from the target are taken. Not all surfaces have normals, which does not cause any problems.

The normals are needed for internal computations in any way. If the surface doesn't have them, they are computed automatically.

Tangents

Tangents are another optional part of the mesh data. They can further be projected or taken from the target.

Lightmap Update

This toggle field is only selectable, if the current *UV2* projection is *Lightmapping*. You may choose if the uv's should be updated after every modification or only as the decals is optimized.

Cast Shadows

Should the meshes produced for this decals instance cast shadows?

Receive Shadows

Should the meshes produced for this decals instance receive shadows?

Use Light Probes

This value is passed to all the mesh renderers that are generated for this decals instance.

Light Probe Anchor

This value is passed to all the mesh renderers that are generated for this decals instance.

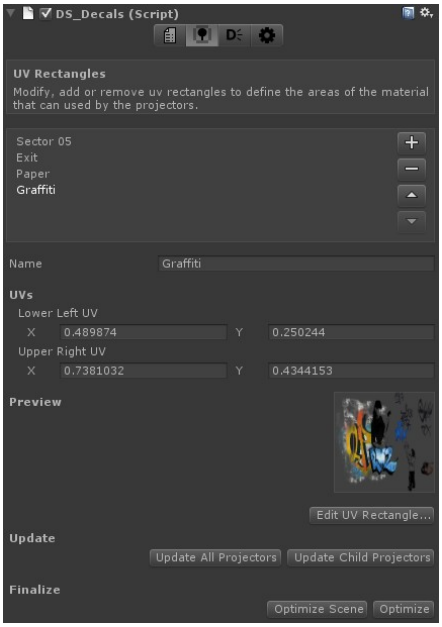
Affect Same LOD Only

Should only renderers be affected by projectors of this decals instance that are in the same LOD as this decals instance itself? More information on that subject can be found in the Level of Detail section.

Mesh Asset Folder

You can specify in which folder the mesh assets of this decals instance have to be stored. This option is only used if the decals belong to a prefab. You find more details including an explanation about the *Create New Mesh Assets* button in the section about Prefabs.

3.3. UV Rectangles Mode



Each decals instance can have as many uv rectangles as needed to define areas for the selected material. That is especially helpful in combination with texture atlases.

First you see a list of the existing uv rectangles. Rectangles can be added by clicking on the plus sign. The minus sign removes the selected uv rectangle. It only works, if it is not used by any projector. The rectangles can be rearranged by moving them up and down.

The properties of the selected rectangle are shown below the list.

Name

The uv rectangle name that can be selected by the projectors in the Decal Projectors Mode.

UVs

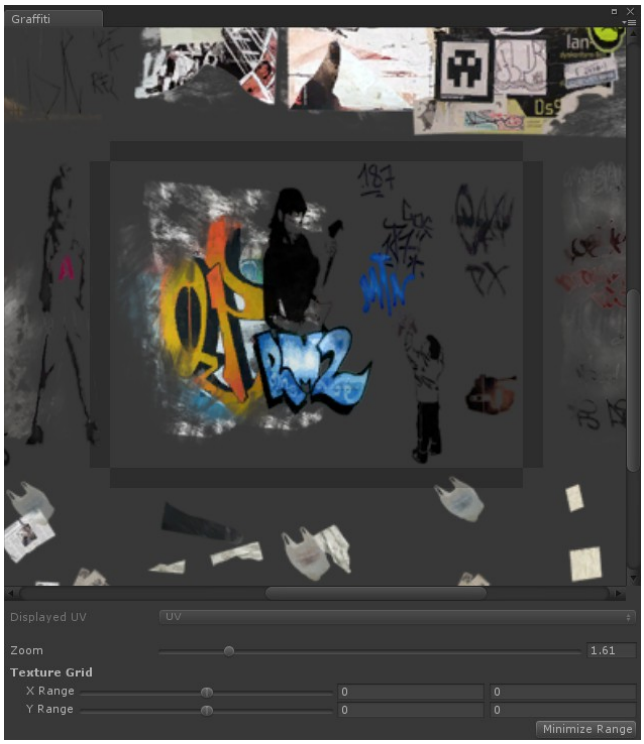
The lower left and upper right uv coordinates can be modified. Though it is more intuitive to do it in the UV Rectangle Window.

Besides the primary uv coordinates for the rectangle, you may modify the secondary ones in the case that they are projected, too.

Preview

The preview shows the uv area of the selected rectangle. The shown texture is always the first one from the selected material. Just below the preview, you find a button to open the UV Rectangle Window.

3.3.1. UV Rectangle Window



This window was created to efficiently modify the uv coordinates of the uv rectangles. It is synchronized with the corresponding view in the inspector, meaning it shows the uv rectangle that is selected in the UV Rectangles Mode's list.

The actual uv rectangle area is surrounded by greyed boxes. The whole area is draggable and each box itself, too.

To drag the scrollable view, use the middle mouse button.

Displayed UV

Depending on which uv's are projected, you may modify the uv coordinates of the rectangle and additionally the uv2 or both coordinates at once.

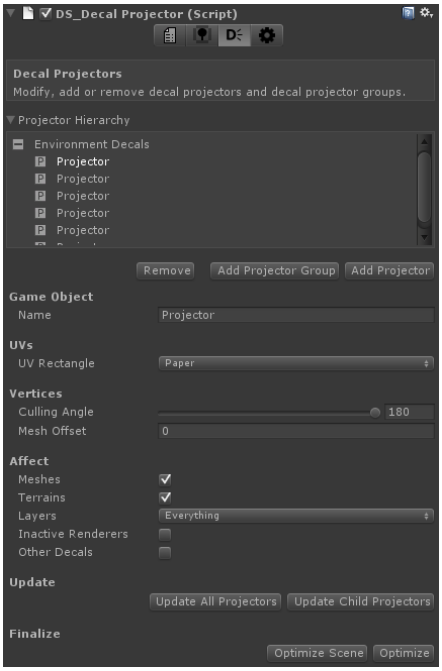
Zoom

Scaling for the texture in the scrolling area.

Texture Grid

Sometime it is necessary to not only operate in the (0, 0) x (1, 1) uv area. This area may be resized by changing the ranges. It is also possible to automatically minimize the range with the button.

3.4. Decal Projectors Mode



This mode looks different depending on the component's type. Decals and projector groups only show the hierarchy and game object name. If a projector is selected, several configurable fields are displayed.

Projector Hierarchy Foldout

This hierarchy contains the decals instance, all groups and projectors. The selection is synchronized with the Unity Editor. You can directly remove and add groups and projectors. Adding one of them means it becomes a child of the currently selected game object.

Name

The name of the currently selected game object.

UV Rectangle

The uv rectangle can be selected among the ones specified in the UV Rectangles Mode.

Culling Angle

Vertices may be culled away. The angle is calculated between the normal of the vertex and the upward direction of the projector. If that angle is greater than the culling angle, the vertex gets removed.

Mesh Offset

Mesh based decals are candidates for z-fighting. Depending on the camera angle, sometimes the decal is shown on top of the mesh onto which it is projected, just like it is supposed to be, but sometimes the decal is drawn below this mesh, which results in a flickering or z-fighting effect.

This effect can be overcome be using a mesh offset which offsets the decal vertices along their normals by the given factor.

If a decals mesh is offset, hard edges become clearly visible in an unwanted way. Hard edges consist of vertices that are at the exact same position, but the normals are pointing in different

direction. As they are offset, the vertices that were at the same position, will not anymore be together. So a closed geometry is splitted slightly.

Affect Meshes

Are meshes affected by this projector? This option is not available for skinned projectors.

Affect Terrains

Are terrains affected by this projector? This option is not available for skinned projectors.

Affected Layers

Which layers are affected by this projector?

Affect Inactive Renderers

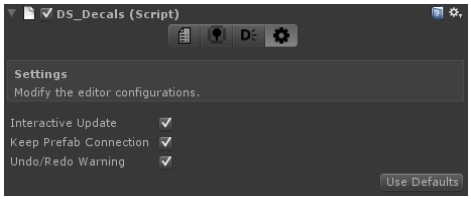
Are deactivated renderers also affected by the projector?

Affect Other Decals

Should other decals also be affected by this projector?

3.5. Setting Mode

The settings for the editor can be made within this mode.



Interactive Update

Should the decals meshes be recalculated after each modification?

Keep Prefab Connection

If projectors and groups are added or removed, the prefab connection is usually lost. With this option enabled, the connection is kept in such a situation.

Undo/Redo Warning

The decals meshes may not be correct if undo or redo operations are performed. Should a warning be shown in this case? If another game object that does not belong to this decals is selected, the warning will disappear even if the decals mesh may still be wrong.

Use Defaults

Revert the settings to the defaults with this button.

4. Advanced Topics

The Decal System is kept as simple as possible for users. For some topics it became necessary to integrate workarounds, such as for prefabs. Other topics that need some more detailed explanations are also covered in this section.

4.1. Prefabs

Decals work in combination with prefabs, but it is necessary to understand some special behaviors you need to consider. They are related to mesh assets. Every model that is imported automatically becomes a prefab and contains one ore more mesh assets. Decals are not different with the exception that the mesh is created in Unity.

Prefab Creation

As you create a prefab that contains a decals instance, you are going to see a warning when you select the prefab, saying that you need to select the prefab instance to create the mesh assets. That's a little tedious, but we could not find a workaround that works in every case. If you select the prefab instance, the mesh assets are automatically created. You may change the directory in which the mesh assets are stored for a better project organization.

Prefab Duplication

If you duplicate a prefab that contains a decals instance, you may want to use the mesh assets from the original prefab. In that case you need to know that any changes that are performed on the mesh of one prefab are automatically present in the other one as well. If that is not the behavior you are looking for, you can use the *Create New Mesh Assets* button in the Decals Mode. This creates new mesh assets for the duplicated prefab. The button can only be pressed if a prefab instance is selected. That issue is also related to the one from the prefab creation and could unfortunately not be resolved.

4.2. Lightmapping

Lightmapping is very simple in Unity as long as the uv's or uv2's of the meshes are correctly prepared. We shortly describe what you need to consider if you want to use lightmapping for the decals.

Decals are like any other models. If a mesh contains only one uv channel, its layout is being used for the lightmapping. If both uv and uv2 are present, the secondary channel's layout is taken for the lightmapping.

Independent of which uv channel is used for the lightmapping, there are two basic rules that have to be followed in order to get correctly baked maps.

1. No overlapping uv areas.
2. All uv coordinates have to be in the in the (0, 0) x (1, 1) range.

There is a simple way to automatically follow those rules. If you import a model to Unity, you have the option *Generate Lightmap UV's*. The Decal System offers exactly the same functionality. Because decals are not imported, but generated directly in Unity, it is slightly different. You have to select *Lightmapping* for the *UV2* projection in the Decals Mode. There are some predefined *Projection*s that are using exactly that setting.

That is the easiest way follow the rules. It is not necessary to use that lightmapping configuration to get it working properly. But you have to make sure on your own that you follow those rules, if you are not using that option.

4.3. Level of Detail

The LOD integration is straight forward. First make sure that the decals instance is in a child game object of the *LODGroup*'s game object that contains the LOD to which the decals instance has to be added. Add the decals instance to that LOD. It makes sense that the projectors only affect the renderers that are in the same LOD, to achieve that, enable *Affect Same LOD Only* which can be found in the Decals Mode.

Be aware that each decals instance can only belong to at most one LOD. It is further not possible to have projectors in different LODs. You need to use different decals for each LOD.

WARNING

Every decals instance can only be used in at most one LOD.

4.4. Shaders

Several shaders are bundled with the Decal System. All are based on the default built-in shaders of Unity. Keep in mind that decals are just a special kind of meshes, so you can equip the materials with the shaders you want.

Transparent Shaders

Transparent shaders cover the whole range from opaque to invisible. Decals that have a material with such a shader do not receive realtime shadows. Shadows are still possible with lightmapping.

Cutout Shaders

Meshes that are rendered with a cutout shader have only fully opaque or invisible parts and nothing in between. They can receive realtime shadows and also support lightmapping.

Mobile Shaders

The non-mobile shaders have a color property to adjust the overall coloring. This option is not present in mobile shaders to get the best possible performance.

Realtime Shadows and Lightmapping Overview

	Diffuse	Bumped	Target Bumped	VertexLit
Transparent	LM	LM	LM*	LM
Mobile Transparent	LM	LM	LM*	-
Cutout	LM, RS	LM, RS	LM*, RS	LM
Mobile Cutout	LM, RS	LM, RS	LM*, RS	-

LM Lightmapping
LM* Lightmapping (uv2 channel must follow the lightmapping rules)
RS Realtime Shadows
– Not implemented

4.5. Skinned Decals

Skinning is a difficult topic for decals in general. Usually they need a quality level that is two times higher than the skinned mesh onto which the projection is made, which is certainly not always available. So make sure that you are using a high quality for the skinned decals. Even if you are using the highest available quality, you may notice imprecisions at certain positions depending on the poses of the model.

WARNING

Use skinned decals with caution!

5. Known Issues

5.1. Terrain LOD

Unity's terrains have built-in level of details. As the camera moves away from the terrain, the geometry is automatically reduced. If decals are placed on the terrain, you will see artifacts due to detail changes.

5.2. Terrain Details and Trees

Decals do not work for terrain details. Particularly grass and trees are not supported. It is hardly possible to create decals for them due to the fact that they can be bended.

5.3. Unprecise Skinned Decals

Skinned Decals may produce inaccurate results. You find more information about that in the Skinned Decals section.

5.4. Texture Leaking

If the scene is saved while a game object is selected which contains a component from the Decal System, there is a warning in the console about leaked textures. That message is harmless and can not be avoided.

5.5. Copying UV Rectangles

It would be useful to be able to copy and paste all uv rectangles from and to a decals instance. If a texture atlas is changed, all the decals that are using it could be updated a lot easier.